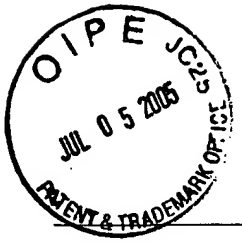
AF



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	
Ryan N. RAKVIC, *et al.*	Examiner: Midys INOA
Serial No: 09/891,523	
Filed: June 27, 2001	Art Unit: 2188
For: PARALLEL CACHELETS	

APPEAL BRIEF UNDER 37 CFR §41.37

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

Sir:

Appellants submit this Appeal Brief in the above-referenced application. A Notice of Appeal was filed on February 4, 2005.

REAL PARTY IN INTEREST

Intel Corporation is the real party in interest for all issues related to this application by virtue of an assignment recorded at Reel 012144, Frame 0523. Carnegie Mellon University also owns an interest in this application.

RELATED APPEALS OR INTERFERENCES

There are no other appeals, interferences, or judicial proceedings known to Appellants, Appellants' legal representative, or assignee which may be related to, directly affect or be directly affected by, or have a bearing on the Board's decision in the pending appeal.

07/06/2005 TLUU11 00000025 110600 09891523
02 FC:1402 500.00 DA

STATUS OF CLAIMS

This application contains claims 9-18, 20-23, 26-28, and 30-33. Claims 9-18, 20-23, 26-28, and 30-33 stand finally rejected and are the subject of this appeal. Claims 1-8, 19, 24, 25, 29, and 34-36 are canceled.

STATUS OF AMENDMENTS

A Response to Final Office Action was filed in this application on November 29, 2004. The Response rebutted the pending rejections and presented new claims directed to the parallel assignment of conflicting and non-conflicting data requests, according to an embodiment of the present invention. An Advisory Action mailed December 23, 2004 states that the Response would not be entered because it presents additional claims without canceling a corresponding number of finally rejected claims. The Advisory Action further states that the Response has been considered but does not place the application in condition for allowance. No other amendments were filed subsequent to the final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

Embodiments of the present invention are directed to a cache and a method for assignment of data requests thereto. Ordinarily, a cache is considered a unified memory that responds to a single read or write request at a time. Embodiments of the present invention, however, propose a cache architecture (element 100) that includes a plurality of parallel cachelets (elements 110-1 through 110-N). Each cachelet therein is individually addressable via $Addr_1$ through $Addr_N$. The cache can respond to multiple requests at a time because each request is directed to one of the cachelets contained therein. If there are four cachelets, for example, the cache 100 can respond to four requests in parallel – each of the four requests is directed to a respective cachelet.

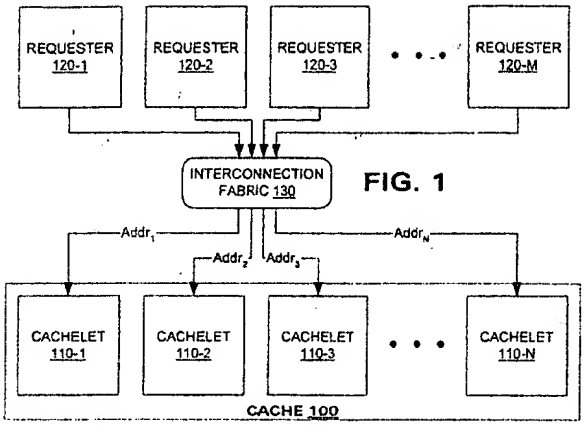


FIG. 1

Requesters (120-1 through 120-M) may provide a cachelet pointer that identifies a cachelet to which their requests are to be directed. The cache provides a method for resolving conflicts when multiple requesters make parallel requests to the same cachelet. In this method, one of the conflicting requests is assigned to the requested cachelet and the remaining conflicting requests are reassigned to unused cachelets in a single clock cycle. In parallel, the cache also assigns non-conflicting requests to their respective requested cachelets.

There are six independent claims involved in this appeal.

Independent claim 10 recites a cache assignment method. (Fig. 3; ¶ 20). According to this method, a cache may receive several data requests, which are associated with cachelet pointers. (¶ 24). The method may then determine whether two or more of the data requests have the same associated cachelet pointer. (¶ 24). If so, the method may present one of the conflicted data requests to the cachelet associated with the cachelet pointer and reassign the remaining conflicted data requests to unused cachelets. (¶ 24). For example, a cache may receive valid data requests L_3 and L_7 during a single clock cycle, where each has cachelet pointer 2, indicating that both should be assigned to cachelet 2. (¶ 24). This produces a conflict for the cache assignment method, which the method may resolve by assigning L_3 to cachelet 2 and assigning L_7 to unused cachelet 1. (¶ 24).

Independent claim 14 recites a cache assignment method. (Fig. 3; ¶ 20). According to this method, a cache may receive several data requests and associated cachelet pointers that address cachelets of the cache. (¶ 24). The method may then determine whether two or more of the data requests have the same associated cachelet pointer. (¶ 24). If so, the method may

present one of the conflicted data requests to the cachelet associated with the cachelet pointer and reassign the remaining conflicted data requests to unused cachelets. (¶ 24). The method may also present data requests that do not have the same cachelet pointers to their identified cachelets. (¶ 24). For example, a cache may receive valid data requests L_3, L_4, and L_7 during a single clock cycle, where L_3 and L_7 have cachelet pointer 2, indicating that both should be assigned to cachelet 2, and L_4 has cachelet pointer 3, indicating that it should be assigned to cachelet 3. (¶ 24). A conflict arises between L_3 and L_7, which the method may resolve by assigning L_3 to cachelet 2 and reassigning L_7 to unused cachelet 1. (¶ 24). Since neither L_3 nor L_7 have the same cachelet pointer as L_4, L_4 may be assigned to its identified cachelet 3. (¶ 24).

Independent claim 16 recites a cache assignment method. (Fig. 3; ¶ 20). According to this method, a cache may receive several data requests and associated cachelet pointers that address cachelets of the cache. (¶ 24). The method may then determine whether any of the cachelet pointers are valid, i.e., whether the cachelet pointers address the data requests to associated cachelets. (¶ 20). If so, the method may forward the data requests with the valid cachelet pointers to the addressed cachelets. (¶ 20). For the remaining data requests that do not have valid cachelet pointers, the method may assign the data requests to unused cachelets according to a default assignment scheme, e.g., round-robin or least-recently-used. (¶ 20). For example, a cache may receive data requests L_3 and L_4 which have been addressed to cachelets 2 and 3, respectively, as indicated by their cachelet pointers. (¶ 24). The cache may receive new data request L_{13} which has not been addressed to a cachelet; hence, L_{13} does not yet have a valid cachelet pointer. (¶ 24). L_3 and L_4 may be forwarded to their respective addressed cachelets 2 and 3. (¶ 24). Whereas, L_{13} may be assigned to the next unused cachelet in a round-robin scheme, e.g., cachelet 0. (¶ 24).

Independent claim 20 recites a cache system. (Figs. 1, 8; ¶¶ 14-16, 42-44). The system may include a cache (element 100, Fig. 1) provided as a first layer (element 610, Fig. 8) of the cache system. (¶¶ 14, 42). The cache may include several independently addressable cachelets (elements 110-1 through 110-N, Fig. 1) and an interconnection fabric (element 130, Fig. 1) that distributes independent loads to the cachelets in a single clock cycle. (¶¶ 15, 16). The system may include a second layer of cache (element 620, Fig. 8) to receive a load that misses the assigned cachelet in the first layer of cache. (¶ 43).

Independent claim 22 recites a cache system, which is substantially similar to that of claim 20. (Figs. 1, 8; ¶¶ 14-16, 42-44).

Independent claim 26 recites a cache assignment method. (Fig. 3; ¶ 20). According to this method, a cache may receive several data requests, which are associated with respective cachelet pointers. (¶ 24). The method may then determine whether any of the cachelet pointers are valid, i.e., whether the cachelet pointers address the data requests to associated cachelets. (¶ 20). If so, the method may forward the data requests with the valid cachelet pointers to the addressed cachelets. (¶ 20). For the remaining data requests that do not have valid cachelet pointers, the method may assign the data requests to unused cachelets according to a default assignment scheme, e.g., round-robin or least-recently-used. (¶ 20). For example, a cache may receive data requests L_3 and L_4 which have been addressed to cachelets 2 and 3, respectively, as indicated by their cachelet pointers. (¶ 24). The cache may receive new data request L_{13} which has not been addressed to a cachelet; hence, L_{13} does not yet have a valid cachelet pointer. (¶ 24). L_3 and L_4 may be forwarded to their respective addressed cachelets 2 and 3. (¶ 24). Whereas, L_{13} may be assigned to the next unused cachelet in a round-robin scheme, e.g., cachelet 0. (¶ 24).

The remaining pending claims depend respectively from these independent claims and recite further limitations therefrom.

GROUNDS OF REJECTION TO BE REVIEWED

The Final Rejection (a) rejects claims 9, 10, 12, 14-18, and 26-28 under 35 U.S.C. §102(e) as anticipated by Rappoport (USP 6,549,987); (b) rejects claims 20-23 under 35 U.S.C. §102(e) as anticipated by Rappoport and The Authoritative Dictionary of IEEE Standards Terms ("Authoritative Dictionary"), where the Authoritative Dictionary is used as an evidentiary reference; and (c) rejects claims 30-33 under 35 U.S.C. §103(a) as being unpatentable over Rappoport. The Final Rejection objects to claims 11 and 13 as being dependent upon a rejected base claim, but allowable if rewritten in independent form.

ARGUMENT

A. The rejections of claims 9, 10, 12, 14-18, and 26-28 as anticipated by Rappoport are improper because Rappoport does not disclose each and every element of the claims.

Claims 9, 10, 12, 14-18, and 26-28 were rejected under 35 U.S.C. §102(e) as anticipated by Rappoport. This rejection is not proper because Rappoport does not disclose each and every element of the claim, as is required for an anticipation rejection. *See* MPEP §2131. In particular, with regard to claims 10 and 14, Rappoport does not disclose, *inter alia*, a method that reassigns data requests with conflicting cachelet pointers to unused cachelets. With regard to claims 16 and 26, Rappoport does not disclose, *inter alia*, a method that, in parallel, forwards data requests with valid cachelet pointers to their respective cachelets and assigns the remaining data requests to unused cachelets according to a default scheme. Each of these claims will be discussed in detail below.

Claim 10 recites "determining whether any of the cachelet pointers conflict with any other cachelet pointer, if a conflict occurs among cachelet pointers, forwarding one of the data requests associated with a conflicting cachelet pointer to the identified cachelet, and reassigning data requests associated with remaining conflicting cachelet pointers to unused cachelets." (Emphasis added.) Rappoport does not disclose this subject matter.

Rappoport discloses a cache and a method for retrieving variable length instruction segments therefrom. FIG.2 of Rappoport, as shown below, is a block diagram of the cache (element 200). An instruction segment is a data object that may be stored among cache banks (not shown) of the segment cache (element 280). Instruction segments have variable lengths and, therefore, portions of a segment may be stored in some or all of the cache banks. If a short instruction segment is read from the segment cache, it may be possible also to retrieve data from another instruction segment that are stored in other cache banks. Rappoport's system identifies and resolves cache bank conflicts to determine whether to retrieve multiple instruction segments in parallel.

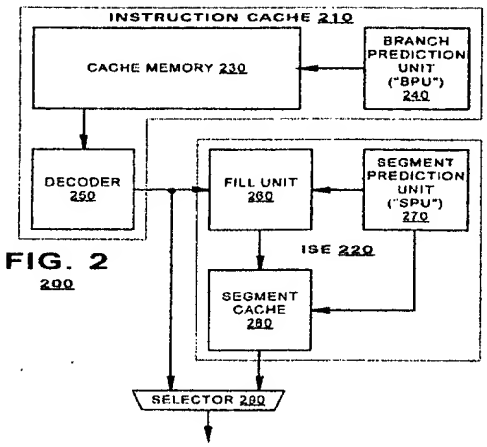


FIG. 2
200

In the Rappoport cache, there is a conflict between cache banks in the segment cache when multiple instruction segments to be retrieved reside in partially overlapping banks. Col. 5, ll. 34-51. The Rappoport cache resolves this conflict by retrieving only the non-conflicting portion of the instruction segment(s) and deferring retrieval of the remaining conflicting portions until at least the next clock cycle. Id. Thus, Rappoport does not teach or suggest reassigning conflicting data requests to unused cachelets. Rappoport's disclosure is clear: When two instruction segments must be read from a common cache bank, the system stalls the read of a second instruction segment and addresses the same cache bank after the read of the first instruction segment concludes. Id. FIG. 6. There is no reassignment.

In response to Appellant's pointing out this distinguishing feature, the Advisory Action mailed December 23, 2004 states:

> Rappoport teaches deferring the remaining data request with the conflicting cachelet pointer; where in deferring the request, the request is being reassigned to the same cachelet which at the time of reassignment is not in use (thus an unused cachelets). (Emphasis added.)

This is an improper claim construction. The Examiner mistakenly equates stalling a memory request to reassigning the memory request By definition, "reassigning" in claim 10 refers to the selection of a different cachelet, not the same cachelet as the Advisory Action erroneously asserts. The assignment to the same bank in Rappoport does not mean the reassignment to a different cachelet, as in claim 10. While claims should be interpreted as broadly as possible, the interpretation here goes far beyond what is proper. Therefore, since Rappoport does not disclose reassigning, as recited in claim 10, claim 10 is not anticipated by Rappoport.

Claim 14 recites "forwarding non-conflicting data requests to a cachelet identified by the cachelet pointer, for the conflicting data requests, forwarding one of the conflicting data requested to the identified cachelet, and reassigning remaining conflicting data requests to unused cachelets." (Emphasis added.) In addition to the reassignment of conflicting data request to unused cachelets, both non-conflicting and conflicting data requests may be assigned to cachelets in parallel. Rappoport does not teach this subject matter. Therefore, claim 14 is also not anticipated by Rappoport.

Claim 16 recites "determining whether any of the cachelet pointers are valid, forwarding data requests having valid cachelet pointers to the addressed cachelet, and assigning remaining data requests to unused cachelets according to a default assignment scheme." (Emphasis added.) Rappoport does not disclose this subject matter. Rappoport clearly does not disclose determining whether any cachelet pointer is valid. Nor does Rappoport disclose assigning data requests with invalid cachelet pointers to unused cachelets. Therefore, claim 16 is not anticipated by Rappoport.

For at least the above reasons with respect to claim 16, claim 26 is also not anticipated by Rappoport.

Claims 9, 12, 15, 17, 18, 27, and 28 are not anticipated by Rappoport for the same reasons based on their dependency from these independent claims. Reversal of these rejections is requested.

B. The rejections of claims 20-23 as anticipated by Rappoport, with the Authoritative Dictionary used as an evidentiary reference, are improper because Rappoport does not disclose each and every element of the claims.

Claims 20-23 were rejected under 35 U.S.C. §102(e) as anticipated by Rappoport. This rejection is not proper because Rappoport does not disclose each and every element of the claim, as is required for an anticipation rejection. *See* MPEP §2131. In particular, with regard to claims 20 and 22, Rappoport does not disclose, *inter alia*, a cache system that distributes its loads to cachelets in a first layer of a cache during a single clock cycle and a second layer of cache to receive a load that misses the cachelet to which it was assigned.

Claim 20 recites "a cache provided as a first layer of the cache system, comprising: a plurality of independently addressable cachelets, means for distributing independent loads to

each of the cachelets in a single clock cycle; and a second layer of cache to receive a load that misses the cachelet to which it was assigned." (Emphasis added.)

In contrast, Rappoport, as stated previously, discloses waiting at least until the next clock cycle to resolve conflicting requests. Col. 5, ll. 47-51. As such, Rappoport does not distribute loads to cachelets in a single clock cycle, as recited in claim 20. Moreover, Rappoport does not teach or suggest a load that misses the cachelet to which it is assigned or a second layer of cache to receive the load, as recited in claim 20.

In response to Appellant's pointing out these distinguishing features, the Advisory Action mailed December 23, 2004 states:

> The system of Rappoport has a first layer of cache (cache memory 230, Figure 2) with cachelets (banks 310, Figure 3) ... and a second layer of cache (segment cache 280, Figure 2) which receives load that misses the cachelet to which it was assigned in that since data is supplied by either the instruction cache or the segment cache (Column 3, lines 4-37) and the selector 290 selects the output from either cache when the first cache memory layer cannot supply the data and thus the load has missed a cachelett (sic) within this cache, the second segment cache (second cache layer) memory layer must receive the missed request.

This assertion is incorrect. Rappoport does not teach or suggest the subject matter of claim 20. In Rappoport, an instruction may be retrieved in segments, which are saved in the segment cache 280. Col. 3, ll. 4-36. Rappoport's instruction segment engine (element 220, FIG. 2 of Rappoport, above) then predicts (element 270) when the instruction segments are to be executed. Id. Only when the instruction segment is to be executed does the selector (element 290) retrieve the instruction segment from the segment cache (element 280). Id. This operation of Rappoport's cache does not include referring the segment prediction to the instruction cache (element 230) if it misses the segment cache. Given that Rappoport's system is directed to segment prediction and retrieval and claim 20 is directed to a cache assignment system, the two are not analogous. Therefore, claim 20 is not anticipated by Rappoport.

For at least the above reasons, claim 22 is also not anticipated by Rappoport.

Claims 21 and 23 are not anticipated by Rappoport for the same reasons based on their dependency from these independent claims. Reversal of these rejections is requested.

C. The rejections of claims 30-33 as being unpatentable over Rappoport are improper because Rappoport does not disclose each and every element of the claims.

Claims 30-33 were rejected under 35 U.S.C. §103(a) as unpatentable over Rappoport. This rejection is not proper because Rappoport does not disclose each and every element of the claims, as required to establish *prima facie* obviousness. *See* MPEP §2143. These claims state:

> 30. The cache assignment method of claim 10, further comprising forwarding the reassigned data requests in parallel with the other forwarded data requests.
>
> 31. The cache assignment method of claim 14, further comprising forwarding the reassigned data requests in parallel with the non-conflicting data requests and the one conflicting data request.
>
> 32. The cache assignment method of claim 14, further comprising forwarding the assigned data requests to the unused cachelets in parallel with the other forwarded data requests.
>
> 33. The cache assignment method of claim 26 further comprising forwarding the assigned data requests in parallel with the forwarding of data requests having valid cachelet pointers.

(Emphasis added.)

As discussed above, Rappoport's system stalls requests when there is a cache bank conflict. A second conflicting request proceeds only after a first conflicting request completes. Rappoport's disclosure describes a sequential operation, which is the exact opposite of a parallel operation such as recited here. The Examiner has provided no explanation why anyone would be motivated to convert Rappoport's sequential operation into a parallel operation. Indeed, because Rappoport's instruction segments are found only in the cache banks that are designated, his operations cannot be converted as the Examiner hypothesizes. Reversal of these rejections is requested.

CONCLUSION

Appellants respectfully request reversal of the rejections of claims 9, 10, 12, 14-18, 20-23, 26-28, and 30-33. These claims, along with allowed claims 11 and 13, are allowable over the cited art.

Respectfully submitted,

Date: July 5, 2005

Cassandra T. Swain, Ph.D.
Registration No. 48,361

KENYON & KENYON
1500 K Street, N.W.
Washington, D.C. 20005
Tel: (202) 220-4200
Fax: (202) 220-4201

OIPE JC25
JUL 05 2005
PATENT & TRADEMARK OFFICE

CLAIMS APPENDIX

1-8. Canceled.

9. The cache assignment method of claim 10, the method further comprising forwarding any data requests associated with non-conflicting cachelet pointers to cachelets identified by the respective pointers.

10. A cache assignment method, comprising:

receiving plural data requests, each associated with respective cachelet pointers,

determining whether any of the cachelet pointers conflict with any other cachelet pointers,

if a conflict occurs among cachelet pointers, forwarding one of the data requests associated with a conflicting cachelet pointer to the identified cachelet, and

reassigning data requests associated with remaining conflicting cachelet pointers to unused cachelets.

11. The cache assignment method of claim 10, wherein multiple data requests having a common set address are forwarded to different cachelets.

12. The cache assignment method of claim 10, further comprising:

determining whether any of the cachelet pointers are valid,

forwarding data requests having valid, non-conflicting cachelet pointers to the addressed cachelet, and

assigning data requests of non-conflicting cachelet pointers to unused cachelets according to a default assignment scheme.

13. The cache assignment method of claim 10, wherein copies of a single data item may be stored in multiple cachelets.

14. A cache assignment method, comprising:

receiving plural data requests and associated cachelet pointers, the cachelet pointers addressing one of a plurality of cachelets within a cache,

determining whether any of the cachelet pointers conflict with any other cachelet pointers,

forwarding non-conflicting data requests to a cachelet identified by the cachelet pointer, for the conflicting data requests, forwarding one of the conflicting data requested to the identified cachelet and

reassigning remaining conflicting data requests to unused cachelets.

15. The cache assignment method of claim 14 wherein the data requests are associated with respective cachelet pointers, the method further comprising:

determining whether any of the cachelet pointers are valid, and

assigning remaining data requests to unused cachelets according to a default assignment scheme.

16. A cache assignment method, comprising:

receiving plural data requests and associated cachelet pointers, the cachelet pointers addressing one of a plurality of cachelets within a cache,

determining whether any of the cachelet pointers are valid,

forwarding data requests having valid cachelet pointers to the addressed cachelet, and

assigning remaining data requests to unused cachelets according to a default assignment scheme.

17. The cache assignment method of claim 16, further comprising:

determining whether any of the cachelet pointers conflict with any other cachelet pointers,

forwarding any data requests associated with non-conflicting cachelet pointers to cachelets identified by the respective pointers.

18. The cache assignment method of claim 16, further comprising:

if a conflict occurs among cachelet pointers, forwarding one of the data requests associated with a conflicting cachelet pointer to the identified cachelet, and

reassigning data requests associated with remaining conflicting cachelet pointers to unused cachelets.

19. Canceled.

20. cache system, comprising:

a cache provided as a first layer of the cache system, comprising:

a plurality of independently addressable cachelets,

means for distributing independent loads to each of the cachelets in a single clock cycle; and

a second layer of cache to receive a load that misses the cachelet to which it was assigned.

21. The cache system of claim 20, wherein the second layer of cache is a system memory.

22. A cache system comprising:

a first layer of cache, comprising a plurality of independently addressable cachelets and means for distributing multiple loads among the cachelets in a single clock cycle, and

a second layer of cache to receive a load that misses the cachelet to which it was assigned.

23. The cache system of claim 22, wherein the second layer of cache is a system memory.

24-25. Canceled.

26. A cache assignment method, comprising:

receiving plural data requests, wherein the data requests are associated with respective cachelet pointers,

determining whether any of the cachelet pointers are valid,

forwarding data requests having valid cachelet pointers to the addressed cachelet, and

assigning remaining data requests to unused cachelets according to a default assignment scheme.

27. The cache assignment method of claim 26, wherein each data request includes a cachelet pointer, the method further comprising forwarding any data requests associated with non-conflicting cachelet pointers to cachelets identified by the respective pointers.

28. The cache assignment method of claim 26, the method further comprising:

receiving plural data requests, each associated with respective cachelet pointers,

determining whether any of the cachelet pointers conflict with any other cachelet pointers,

if a conflict occurs among cachelet pointers, forwarding one of the data requests associated with a conflicting cachelet pointer to the identified cachelet, and

reassigning data requests associated with remaining conflicting cachelet pointers to unused cachelets.

29. Canceled.

30. The cache assignment method of claim 10, further comprising forwarding the reassigned data requests in parallel with the other forwarded data requests.

31. The cache assignment method of claim 14, further comprising forwarding the reassigned data requests in parallel with the non-conflicting data requests and the one conflicting data request.

32. The cache assignment method of claim 14, further comprising forwarding the assigned data requests to the unused cachelets in parallel with the other forwarded data requests.

33. The cache assignment method of claim 26 further comprising forwarding the assigned data requests in parallel with the forwarding of data requests having valid cachelet pointers.

34-36. Canceled.

EVIDENCE APPENDIX

There is no evidence provided pursuant to Rule 1.130, 1.131, or 1.132, or any evidence entered by the Examiner and relied upon by Appellants.

RELATED APPEALS APPENDIX

There are no other appeals, interferences, or judicial proceedings known to Appellants, Appellants' legal representative, or assignee which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

OIPE JC25
JUL 05 2005
PATENT & TRADEMARK OFFICE

PTO/SB/17 (10-04v2)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2005

Effective 10/01/2004. Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 500.00

Complete if Known	
Application Number	09/891,523
Filing Date	June 27, 2001
First Named Inventor	R. RAKVIC, et al.
Examiner Name	Midys Inoa
Art Unit	2188
Attorney Docket No.	2207/1123601

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None

☒ Deposit Account:

Deposit Account Number: 11-0600

Deposit Account Name: Kenyon & Kenyon

The Director is authorized to: *(check all that apply)*
☒ Charge fee(s) indicated below ☒ Credit any overpayments
☒ Charge any additional fee(s) or any underpayment of fee(s)
☒ Charge fee(s) indicated below, **except for the filing fee** to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	790	2001	395	Utility filing fee	
1002	350	2002	175	Design filing fee	
1003	550	2003	275	Plant filing fee	
1004	790	2004	395	Reissue filing fee	
1005	160	2005	80	Provisional filling fee	

SUBTOTAL (1) ($) 0

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

			Extra Claims		Fee from below		Fee Paid
Total Claims		-20 **	=	X	18	=	
Independent Claims		-3 **	=	X	88	=	
Multiple Dependent				X		=	0

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	88	2201	44	Independent claims in excess of 3
1203	300	2203	150	Multiple dependent claim, if not paid
1204	88	2204	44	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($)

**or number previously paid, if greater; For Reissues, see above*

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet.	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	120	2251	60	Extension for reply within first month	
1252	450	2252	2225	Extension for reply within second month	
1253	1,020	2253	510	Extension for reply within third month	
1254	1,590	2254	795	Extension for reply within fourth month	
1255	2,160	2255	1,080	Extension for reply within fifth month	
1401	500	2401	250	Notice of Appeal	
1402	500	2402	225	Filing a brief in support of an appeal	500
1403	300	2403	150	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive – unavoidable	
1453	1,370	2453	685	Petition to revive – unintentional	
1501	1,370	2501	685	Utility issue fee (or reissue)	
1502	490	2502	245	Design issue fee	
1503	660	2503	330	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17 (q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	790	2809	395	Filing a submission after final rejection (37 CFR § 1.129(a))	
1810	790	2810	395	For each additional invention to be examined (37 CFR § 1.129(b))	
1801	790	2801	395	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 500

SUBMITTED BY

Name (Print/Type)	Cassandra T. Swain, Ph.D.	*Registration No. (Attorney/Agent)*	48,361	*Telephone*	202.220.4200
Signature	Cassandra T. Swain			*Date*	July 5, 2005

Complete (if applicable)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

This collection of information is required by 37 CFR 1.17 and 1.27. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**